SEO en diseño y maquetación

SEO, diseño, maquetación, HTML, CSS, Javascript

Los buscadores entienden los sitios

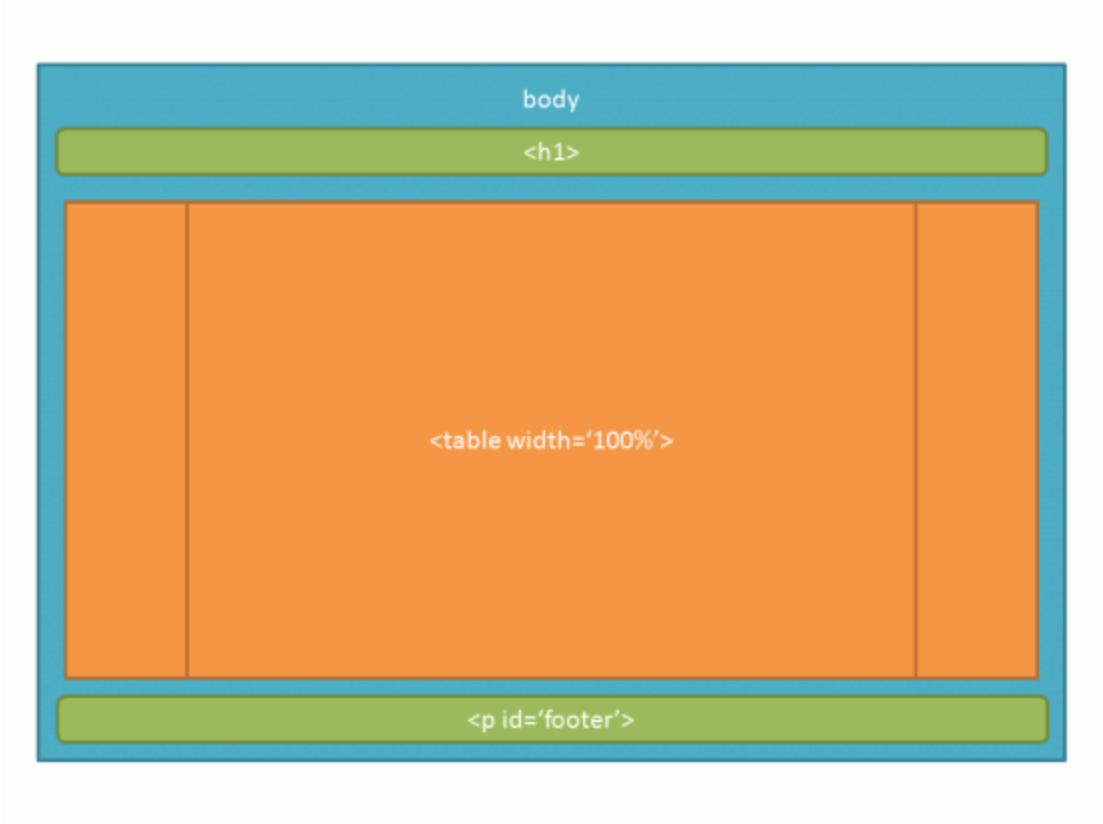
- El diseño es importante, pero es totalmente imprescindible.
- Se posiciona mejor una página en HTML plano, sin diseño que una página con diseño.
- Todos los sitios deben tener diseño porque permiten que el usuario navegue más, se siente más cómodo y consuma la información del sitio.
- Es importante que sea usable e indexable.
- Hay que maquetar de forma que se aplique la menor cantidad de código posible.

La estructura

- Los sitios deben tener una misma estructura a lo largo de todo el sitio, pudiendo hacer una excepción en la página principal.
- La estructura ideal es aquella en la que hay:
  - Cabecera
  - Menú de navegación (único)
  - Camino de navegación
  - Contenido
  - Zona complementaria al contenido
  - Pie de página (muy limitado)

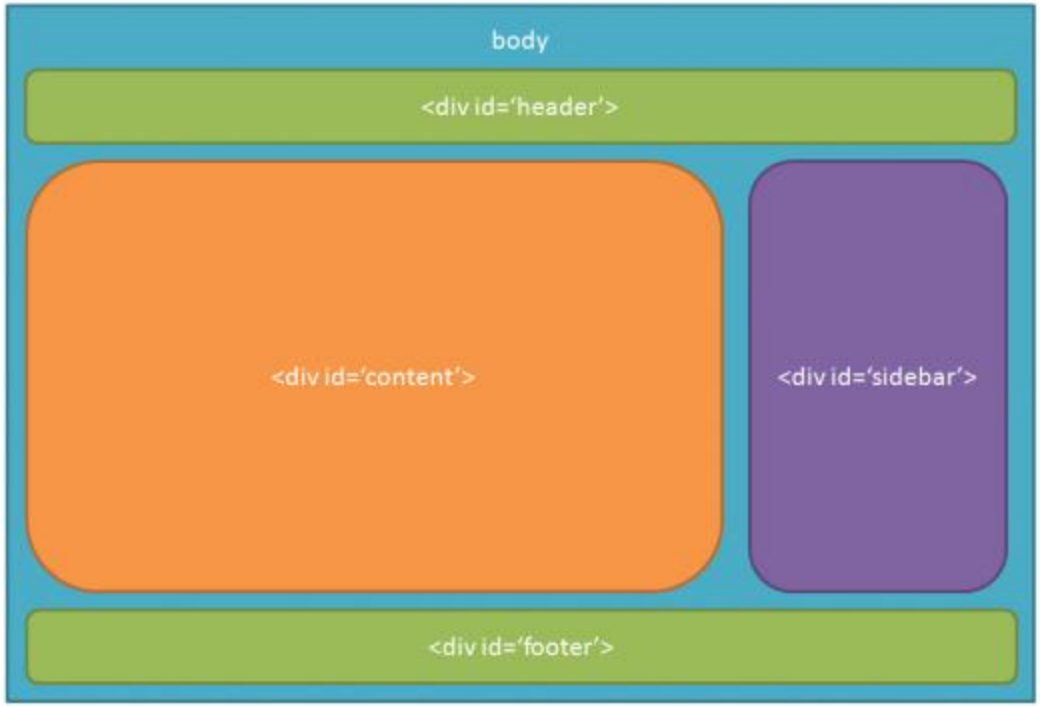
El formato de las páginas

- Estructura básica con HTML 3



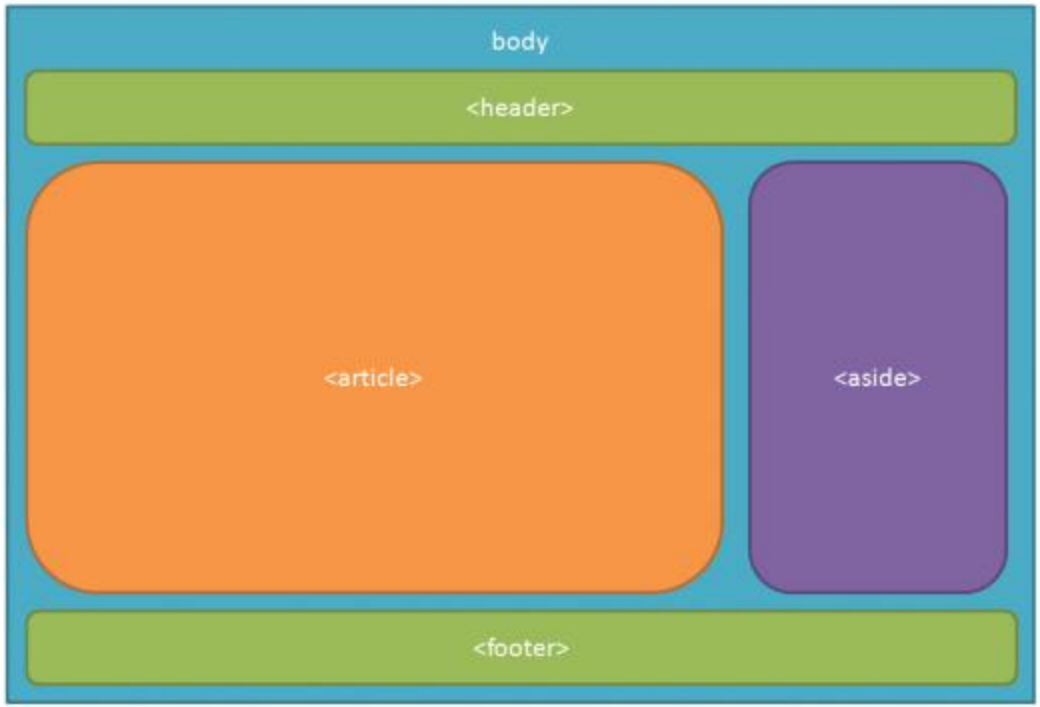
El formato de las páginas

- Estructura básica con HTML 4



El formato de las páginas

- Estructura básica con HTML 5



las nuevas etiquetas contenedoras

- <header>
  - La inicio de los distintos elementos...
- <footer>
  - El final de los distintos elementos
- <article>
  - Lo más importante de la página (o sea, lo que hay que «posicionar»).
- <aside>
  - Los contenidos relacionados con el contenido principal (article)

las nuevas etiquetas contenedoras

- <section>
  - Las diferentes secciones con «identidad propia» donde organizar contenidos.
- <nav>
  - El bloque que agrupa los enlaces
- <figure>
  - El bloque que agrupa imágenes, elementos multimedia...

Creando con rendimiento

- **Rapidez por defecto**: muchas aplicaciones que se construyen para CMS, lenguajes de programación, “la nube”, bibliotecas de JavaScript, navegadores, servidores… ya están pensadas para ir rápido.

- **Maquetación del navegador**: con el fin de hacer que las páginas web más rápido los desarrolladores necesitan la capacidad de encontrar qué partes son más lentas. Esto requiere revisar el tiempo que tarda en cargar y ejecutarse el JavaScript, los CSS, la maquetación de los elementos, la gestión del DOM…

- **Consolidación**: las herramientas de rendimiento de la web, servicios y similares no han llevado un único camino, sino que cada uno ha puesto sus esfuerzos de forma separada. Eso va a cambiar y pronto veremos herramientas que combinan la depuración de JavaScript, el perfil de JavaScript, DOM, el uso de la red… todo en una sola herramienta. Las métricas de rendimiento se gestionarán desde un único panel en lugar de tener que visitar múltiples servicios separados. La consolidación también va a ocurrir a nivel de empresa, donde las empresas más pequeñas relacionados con el rendimiento son adquiridos por las grandes empresas de consultoría y servicios.

Creando con rendimiento

- **TCP y HTTP**: Los protocolos por los que funciona Internet deben ser optimizados, y SPDY es una propuesta. Tenemos que tratar de conseguir más apoyo para el *pipelining*. Cualquier mejora en la red llegará a todos los sitios y usuarios.

- **Estándar**: hay que establecer un estándar sobre las formas de medir, los datos, las pruebas… La Web Timing Spec es un primer ejemplo a tener presente.

- **Organizaciones en la industria**: dentro del mundillo de la WPO veremos nacer y crecer organizaciones profesionales, formación, certificaciones, organismos de normalización… Un ejemplo podría ser que los editores web compartan información acerca de los anuncios de publicidad lentos.

- **Los datos**: hacer seguimiento de los resultados y encontrar nuevas oportunidades de rendimiento requiere un gran análisis de datos. Es probable que comiencen a verse repositorios públicos de datos relacionados con el rendimiento.

Creando con rendimiento

- **Verde**: los estudios realizados que cuantifican cómo mejorar el funcionamiento web confirman la reducción del consumo de energía y por ello la contaminación que generan los centros de datos.

- **Rendimiento móvil**: es como un nuevo punto de partida, se necesita recopilar todo tipo de información hasta encontrar los principales problemas, las causas y encontrar soluciones y crear herramientas para así poder ofrecer información sobre todo esto.

- **La velocidad como elemento diferenciador**: muchas de las decisiones que se tomarán sobre Internet se basarán en el rendimiento. Cuando alguien adquiera un dispositivo, elija un proveedor, se revise un sitio web, la lealtad de los usuarios será un factor importante a la hora de hacer mediciones.

Qué espera un usuario

- El 47% de los usuarios esperan que una página cargue en menos de 2 segundos.
- El 14% cambia de tienda online si la página tarda en cargar.
- El 40% de los usuarios abandona una página que tarda más de 3 segundos en cargar.
- El 64% de los compradores que no están satisfechos cambia de sitio para su próxima compra.
- El 52% de los compradores afirman que un sitio que carga rápido los fideliza.

NAVIGATIONTIMING, el estándar del W3C

- El W3C (organismo que vela por los estándares de Internet) creó hace un tiempo un grupo de trabajo para crear un estándar en lo que a tiempos de carga se refiere.
- Aunque la propuesta no es definitiva, ya existe una primera versión del estándar que permite ver algunos datos de la propuesta. Ahora mismo funciona en Internet Explorer 9 y en Google Chrome 11. Puedes acceder a esta página para ponerlo a prueba.
- Este estándar tiene una serie de atributos con los que podremos tener ciertos datos:
- ***navigationStart***:
  El valor corresponde al tiempo inmediatamente después que el navegador hace un "unload" del documento anterior. En caso de que no haya documento anterior, devolverá lo mismo que *fetchStart*. Si existen redirecciones anteriores o no hay origen, el valor será 0.
- ***unloadEventStart***:
  Si el documento anterior y el actual tienen el mismo origen se devolverá el valor que haya justo antes de ejecutar el "unload". Si no hay documento previo o es una redirección, será 0.

NAVIGATIONTIMING, el estándar del W3C

- ***unloadEventEnd***:
  Si el documento anterior y el actual tienen el mismo origen, se devolverá el valor que haya justo después de ejecutar el "unload". Si no hay documento previo, o el "unload" no se lleva a cabo correctamente se devolverá 0.
- ***redirectStart***:
  Si hay redirecciones y son del mismo origen, se devolverá el momento de inicio en el que se inicia la redirección. Sino estará a 0.
- ***redirectEnd***:
  Si hay redirecciones y son del mismo origen, se devolverá el momento en el que se reciba el último byte de la última redirección. Sino estará a 0.
- ***fetchStart***:
  Si el recurso que se va a mostrar se llama mediante una consulta GET, se devolverá el valor anterior de que el navegador compruebe si existe algún tipo de caché del recurso. En cualquier otro caso devolverá el valor en el que el navegador comience a recuperar el recurso.

NAVIGATIONTIMING, el estándar del W3C

- ***domainLookupStart***:
  Se devolverá el valor inmediatamente anterior a que el navegador haga la petición DNS del documento. En el caso de conexiones persistentes o que esté cacheado, devolverá el mismo valor que *fetchStart*.
- ***domainLookupEnd***:
- Se devolverá el valor inmediatamente después a que el navegador haya realizado la petición DNS y haya sido contestada. En el caso de conexiones persistentes o que esté cacheado, devolverá el mismo valor que *fetchStart*.
- ***connectStart***:
  Devuelve el momento inmediatamente anterior a que el navegador haga la petición para recuperar el documento. En el caso de que existan cachés o conexiones persistentes se devolverá el mismo valor de *domainLookupEnd*.
- ***connectEnd***:
  Devuelve el momento inmediatamente después a que el navegador acabe la petición para recuperar el documento actual. En el caso de que existan cachés o conexiones persistentes se devolverá el mismo valor de *domainLookupEnd*. En el caso de que haya un error en la conexión, se devolverá los valores de la nueva conexión. Este valor también incluirá la conexión en el protocolo SSL.

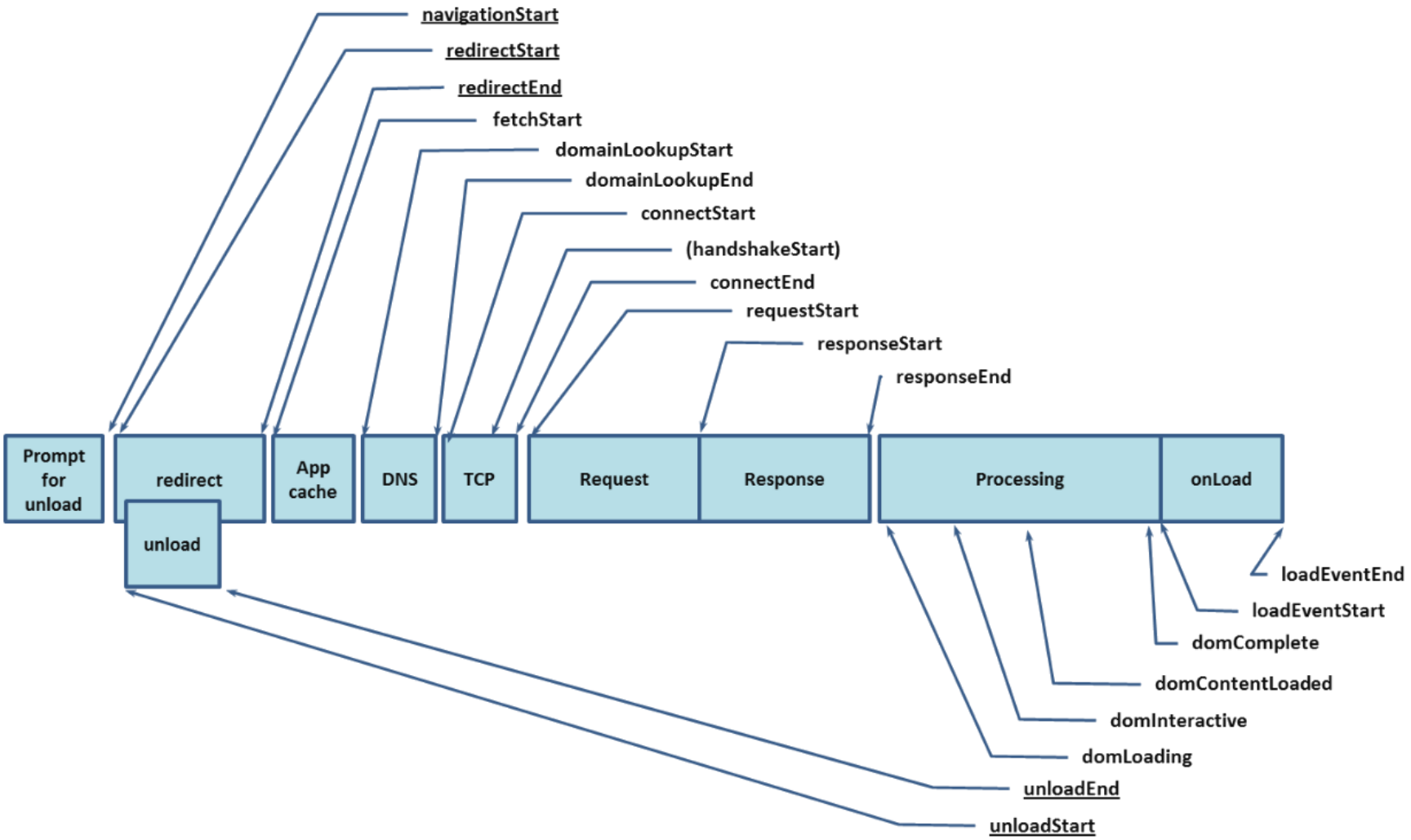
NAVIGATIONTIMING, el estándar del W3C

- ***secureConnectionStart***:
  (Opcional). Devuelve el tiempo inmediatamente anterior a que el navegador comience a procesar la petición de protocolo seguro. Si no se utiliza, devolverá 0.
- ***requestStart***:
  Devuelve el momento inmediatamente anterior a que se solicite el documento actual. Es aplicado antes de comprobar las cachés. Si hay un error en la conexión, se devuelve el momento en el que se inicia la nueva petición.
- ***responseStart***:
  Devuelve el tiempo inmediatamente anterior a que el navegador reciba el primer byte de información desde el servidor o desde la caché.
- ***responseEnd***:
  Devuelve el tiempo inmediatamente posterior a que el navegador reciba el último byte del documento o se cierre la conexión.
- ***domLoading***:
  El momento en el que el navegador comienza el “loading” de la página.
- ***domInteractive***
  El momento en el que el navegador permite la “interactive” de la página.

NAVIGATIONTIMING, el estándar del W3C

- ***domContentLoadedEventStart***:
  Devuelve el instante en el que el navegador establece el evento *DOMContentLoaded*.
- ***domContentLoadedEventEnd***:
  Devuelve el instante en el que el navegador finaliza el evento *DOMContentLoaded*.
- ***domComplete***:
  El momento en el que el navegador marca el “complete” de la página.
- ***loadEventStart***:
  Devuelve el tiempo en el que el documento se dispara, quedando a 0 mientras esto no ocurra.
- ***loadEventEnd***:
  Devuelve el tiempo en el que el documento está completo, quedando a 0 mientras esto no ocurra o sea erróneo.

NAVIGATIONTIMING, el estándar del W3C



Cosas que hay que hacer desde el principio

- Combinar CSS y JavaScript en un único fichero
- Combinar imágenes / iconos en un CSS Sprite
- Incluir los CSS en la parte superior
- Incluir los JavaScript en la parte inferior
  - Usar «defer» en los que no tengan *document.write*
  - Usar «async» en los que sean simples funciones

```
<script src="fichero.js" defer></script>
<script src="fichero.js" async></script>
```

Cosas que hay que hacer desde el principio

- Posible forma de cargar un JavaScript

```
var js = document.createElement("script");
js.preload = true;
js.src = "fichero.js"; //aquí se descarga
js.onpreload = function() {
        document.body.appendChild(script); //aquí se ejecuta
};
```

- Posible forma de cargar un CSS

```
var h = document.getElementsByTagName('head')[0];
var link = document.createElement('link');
link.href = 'fichero.css';
link.type = 'text/css';
link.rel = 'stylesheet';
h.appendChild(link);
```

Cosas que hay que hacer desde el principio

- Externalizar los JavaScript y CSS a ficheros externos.
- Minimizar los ficheros JavaScript y CSS.
- Generar un fichero úncio y cachearlo en un servidor de estáticos.
- Anticipar y cachear elementos futuros.
  - Uso del «prefetch»
- Reducir las peticiones DNS (no llamar a dominios de terceros).

```
<link rel="dns-prefetch" href="http://www.example.com/">
```

Optimización de imágenes

- Reducir al máximo la cantidad de colores de uso en la paleta
  - Usar algún software como ImageMagick o Smush.it.
- Reducir el tamaño de las imágenes
  - Intentar convertir los GIF en PNG si reducen su tamaño; por lo general hacer esto suele reducir el tamaño de los archivos cerca de un 20%, por ejemplo con ImageMagick.
  - Optimizar los GIF animados; una de las herramientas que permite hacerlo es Gifsicle.
  - Optimizar los PNG; puede llegarse a reducir sobre un 15% con herramientas como Pngcrush.
  - Optimizar los JPG; puede reducirse el tamaño cerca de un 10% con herramientas como jpegtran.

Optimización de imágenes

- Las imágenes han de usarse en su tamaño exacto (no se pueden escalar).
- En el elemento <img> hay que indicar de alguna manera (vía CSS o el propio «width» y «height») el tamaño.
- Los CSS Sprites deberían ser PNG 8 (o sea, 8 bits o 256 colores)

Herramientas para optimizar imágenes

- **Herramientas en línea:**
  - Online Image Optimizer <*http://tools.dynamicdrive.com/imageoptimizer/*>
  - PunyPNG <*http://www.punypng.com/*>
  - SiteReportCard Image Optimization <*http://sitereportcard.com/imagereducer.php*>
  - Smush.it <*http://www.smushit.com/*>

- **Herramientas de escritorio:**
  - ImageOptim <*http://imageoptim.pornel.net/*> - Mac OSX
  - JPG & PNG Stripper <*http://www.steelbytes.com/?mid=30*> - Windows
  - OptiPNG <*http://optipng.sourceforge.net/*>
  - PNGGauntlet <*http://benhollis.net/software/pnggauntlet/*> - Windows
  - PNGOUT <*http://www.advsys.net/ken/util/pngout.htm*> - Windows
  - Radical Image Optimization Tool <*http://luci.criosweb.ro/riot/*> - Windows
  - Shrink O'Matic <*http://toki-woki.net/p/Shrink-O-Matic/*> - Adobe AIR
  - TweakPNG <*http://entropymine.com/jason/tweakpng/*> - Windows

Optimización de Flash

- Aunque los ficheros de Adobe Flash no son concretamente imágenes, sí que es cierto que como elementos multimedia se deben un respeto, ya que existen en muchos sitios web.
- Los ficheros SWF están comprimidos de por sí gracias al sistema deflate propio de Java, por lo que al final, lo que recibimos ya está reducido, pero se puede reducir aún más gracias a 7-Zip, que comprime mejor.
- Para ello usaremos apparat. Es recomendable mirar en profundidad su configuración.

Optimización de multimedia

```
reducer -i test\test.swf
[i] Apparat -- http://apparat.googlecode.com/
[i] Launching tool: Reducer
[i] Waiting for 7z ...
[i] Compression ratio: 18.224573%
[i] Total bytes: 310
[i] Completed in 547ms.
```

Elementos DOM

- Los elementos DOM de una página web son prácticamente todos aquellos elementos con los que se puede interactuar de alguna manera. Básicamente se podría resumir en que son todos los tags, ID, clases, etc...

- Hay que mantener una cantidad reducida de elementos del DOM (por debajo de los 1.000)

- Una forma de saber cuántos elementos DOM estamos utilizando es ejecutar en la consola de Firebug la siguiente consulta:

```
document.getElementsByTagName('*').length
```

Elementos DOM

- Reducir los elementos «iframe»
- Usar tablas con el ancho fijo
  - Establecer el elemento CSS *table-layout* a *fixed* en la tabla.
  - Definir de forma explícita los *col* para cada columna.
  - Establecer el atributo *width* para cada columna.
- Cerrar los elementos del HTML

Gracias

Javier Casares
javier.casares@kisslab.com - http://javiercasares.com